UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/081,079	02/21/2002	Rolf Gunter Erich Stegelmann	10157	2341

26890 7590 12/07/2004

JAMES M. STOVER
NCR CORPORATION
1700 SOUTH PATTERSON BLVD, WHQ4
DAYTON, OH 45479

EXAMINER	
WASSUM, LUKE S	

ART UNIT	PAPER NUMBER
2167	

DATE MAILED: 12/07/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/081,079	STEGELMANN ET AL.	
	Examiner	Art Unit	
	Luke S. Wassum	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 September 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *19 June 2002* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Response to Amendment

1. The Applicants' amendment, filed 2 September 2004, has been received, entered into the record, and considered.

2. As a result of the amendment, claims 1, 7, 9, 10, 12-14 and 17 have been amended. Claims 1-29 remain pending in the application.

The Invention

3. The claimed invention is a database system including components for executing stored procedures that include at least one of a conditional expression, assignment expression, and dynamic database query language (e.g., SQL) statement, wherein low-level code is generated for each such expression, and inserted into the object code of the corresponding stored procedure.

Claim Rejections - 35 USC § 112

4. In view of the Applicants' amendments to claims 14 and 17, the examiner withdraws the pending claim rejections under 35 U.S.C. § 112.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. The factual inquiries set forth in *Graham* v. *John Deere Co*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

7. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

8. Claims 1-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Chow et al.** (U.S. Patent 5,875,334) in view of **Oracle** ("PL/SQL User's Guide and Reference").

9. Regarding claim 1, **Chow et al.** teaches a method of executing a stored procedure in a database system as claimed, the stored procedure containing at least an expression and a database query language statement, the method comprising:

a) identifying the expression in the stored procedure, the expression being according to one of plural predetermined types of expressions (see col. 8, lines 7-10);

b) generating low-level code representing the expression (see col. 8, lines 18-27); and

c) generating an object representing the stored procedure, the object containing the low-level code and one or more instructions representing the database query language statement (see col. 8, lines 27-32).

Chow et al. does not explicitly teach a method wherein the low-level code comprises object code.

Oracle, however, teaches a method wherein stored procedures, embodied in PL/SQL language routines, are compiled for native execution, requiring the generation of object code to represent the stored procedure (see pages 8-9 Compiling PL/SQL Code for Native Execution).

It would have been obvious to one of ordinary skill in the art at the time of the invention to compile stored procedures into object code, since execution of stored procedures will be faster if they are compiled into native code (see pages 8-9 Compiling PL/SQL Code for Native Execution).

10. Regarding claim 15, **Chow et al.** teaches an article comprising at least one storage medium containing software as claimed, that when executed cause a database system to:

a) generate low-level code corresponding to a stored procedure having at least a first type expression and a second type expression, the first type expression selected from the group consisting of a conditional expression, an assignment expression and a dynamic database query language statement (see col. 8, lines 7-27);

b) create a predetermined type of code corresponding to the first type expression (see col. 8, lines 7-27);

c) provide the predetermined type of code in the low-level code to represent the first type expression (see col. 8, lines 7-27); and

d) provide one or more instructions representing the second type expression in the low-level code, the instructions different from the predetermined type of code (see col. 8, lines 27-32).

Chow et al. does not explicitly teach a method wherein the low-level code comprises object code.

Oracle, however, teaches a method wherein stored procedures, embodied in PL/SQL language routines, are compiled for native execution, requiring the generation of object code to represent the stored procedure (see pages 8-9 Compiling PL/SQL Code for Native Execution).

It would have been obvious to one of ordinary skill in the art at the time of the invention to compile stored procedures into object code, since execution of stored procedures will be faster if they are compiled into native code (see pages 8-9 Compiling PL/SQL Code for Native Execution).

11. Regarding claim 23, **Chow et al.** teaches a database system as claimed, comprising:

a) a plurality of nodes (see disclosure of client and server, col. 3, lines 15-19);

b) an evaluator module in a first one of the plurality of nodes (see disclosure that the stored procedure is invoked from the client, col. 4, lines 10-16);

c) an access module in a second one of the plurality of nodes, the access module to manage access to a portion of data stored in the database system (see disclosure that the statement is executed on the server, col. 3, lines 15-19); and

d) a controller in the first node adapted to execute a stored procedure low-level code, the low-level code containing a first type of code to represent an expression that is one of a conditional expression, an assignment expression and a dynamic statement, the low-level code containing a second, different type of code to represent a database query language statement (see col. 8, lines 7-27),

e) the controller adapted to submit the first type of code to the evaluator module to evaluate the expression (see col. 8, lines 7-27; see also disclosure of the separate PGM pointer for the 'procedural part' and the pointer to the executable plan for the query statement, col. 31, line 22 through col. 32, line 2); and

f) the controller adapted to submit a command corresponding to the database query language statement to the access module (see col. 3, lines 15-21; see also disclosure of

the separate PGM pointer for the 'procedural part' and the pointer to the executable plan for the query statement, col. 31, line 22 through col. 32, line 2).

Chow et al. does not explicitly teach a method wherein the low-level code comprises object code.

Oracle, however, teaches a method wherein stored procedures, embodied in PL/SQL language routines, are compiled for native execution, requiring the generation of object code to represent the stored procedure (see pages 8-9 Compiling PL/SQL Code for Native Execution).

It would have been obvious to one of ordinary skill in the art at the time of the invention to compile stored procedures into object code, since execution of stored procedures will be faster if they are compiled into native code (see pages 8-9 Compiling PL/SQL Code for Native Execution).

12. Regarding claim 27, **Chow et al.** teaches an article comprising at least one storage medium containing instructions for use in a database system as claimed, the instructions when executed causing the database system to:

a) access low-level code in response to invocation of a stored procedure, the low-level code containing first type of code representing an expression and a second type code representing a database query language statement (see col. 8, lines 7-27);

b) submit the first type code to an evaluator module to evaluate the expression (see col. 8, lines 7-27); and

c) submit a command corresponding to the database query language statement to an access module to access data specified by the database query language statement (see col. 3, lines 15-21).

Chow et al. does not explicitly teach a method wherein the low-level code comprises object code.

Oracle, however, teaches a method wherein stored procedures, embodied in PL/SQL language routines, are compiled for native execution, requiring the generation of object code to represent the stored procedure (see pages 8-9 Compiling PL/SQL Code for Native Execution).

It would have been obvious to one of ordinary skill in the art at the time of the invention to compile stored procedures into object code, since execution of stored procedures will be faster if they are compiled into native code (see pages 8-9 Compiling PL/SQL Code for Native Execution).

13. Regarding claims 2 and 16, **Chow et al.** additionally teaches a method and article further comprising directly executing low-level code at run-time to evaluate the expression (see col. 27, lines 14-22).

14. Regarding claim 3, **Chow et al.** additionally teaches a method wherein directly executing the low-level code is performed in place of submitting a database query language statement to evaluate the expression (see col. 8, lines 7-27).

15. Regarding claim 4, **Chow et al.** additionally teaches a method wherein directly executing the low-level code consumes less database system resources than submitting a database query language statement to evaluate the expression (see col. 8, lines 42-48).

16. Regarding claims 5 and 18, **Chow et al.** additionally teaches a method and article further comprising:

a) submitting the low-level code to an evaluator module to evaluate the expression (see col. 8, lines 7-27); and

b) submitting a command corresponding to an access module in the database system to access data specified by the database query language statement (see col. 3, lines 15-21).

17. Regarding claim 6, **Chow et al.** additionally teaches a method wherein the database system has a first node containing a parsing engine and the evaluator module (see col. 8, lines 7-27) and a second node containing the access module (see disclosure that the statement is executed on the server, col. 3, lines 15-19), wherein submitting the command is performed by the parsing engine (see col. 8, lines 7-27).

18. Regarding claim 7, **Chow et al.** additionally teaches a method further comprising:

a) storing information pertaining to a variable and a constant used in the expression with the low-level code in the object (see col. 18, lines 25-60);

b) executing the low-level code during execution of the stored procedure using an evaluator module (see col. 27, lines 58-60); and

c) using the information pertaining to the variable and constant during execution of the low-level code to evaluate the expression (see col. 30, lines 27-52).

19. Regarding claim 8, **Chow et al.** additionally teaches a method wherein identifying the expression comprises identifying one of a conditional expression, an assignment expression and a dynamic database query language statement (see col. 8, lines 7-27).

20. Regarding claim 9, **Chow et al.** additionally teaches a method further comprising:

a) identifying a second expression in the stored procedure that is one of a conditional expression, an assignment expression, and a dynamic database query language expression (see col. 8, lines 7-27); and

b) generating second low-level code to represent the second expression, wherein generating the object comprises providing the second low-level code in the object (see col. 8, lines 7-27).

21. Regarding claim 10, **Chow et al.** additionally teaches a method wherein generating the object comprises generating the object containing the low-level code that is different from the instructions representing the database query language statement (see col. 7, lines 50-65).

22. Regarding claim 11, **Chow et al.** additionally teaches a method wherein generating the low-level code comprises generating assembly code (see disclosure that the procedural and non-procedural parts of the stored procedure are run through compilers, col. 7, line 66 through col. 8, line 6).

23. Regarding claim 12, **Chow et al.** additionally teaches a method further comprising:

a) storing the object in a predetermined location (see col. 4, lines 8-15); and

b) accessing the predetermined location to retrieve the object in response to invocation of the stored procedure (see col. 4, lines 8-15).

24. Regarding claim 13, **Chow et al.** additionally teaches a method wherein storing the object in the predetermined location comprises storing the object in a stored procedure table (see col. 4, lines 8-15).

25. Regarding claims 14 and 20, **Chow et al.** additionally teaches a method and article further comprising:

a) executing the object, wherein executing the object comprises submitting the low-level code to an evaluator module to execute the low-level code wherein the database query language statement is not one of the predetermined types of expressions (see disclosure of the separate PGM pointer for the 'procedural part' and the pointer to the executable plan for the query statement, col. 31, line 22 through col. 32, line 2); and

b) executing the one or more instructions representing the database query language statement without submitting the one or more instructions to the evaluator module (see disclosure of the separate PGM pointer for the 'procedural part' and the pointer to the executable plan for the query statement, col. 31, line 22 through col. 32, line 2).

26. Regarding claim 17, **Chow et al.** additionally teaches an article wherein the software when executed causes the database system to submit a database query language statement in the second type expression to the database system to evaluate the second type expression (see col. 3, lines 8-35).

27. Regarding claim 19, **Chow et al.** additionally teaches an article wherein the software when executed causes the database system to provide the predetermined type of code to the low-level code generator to add the low-level code (see col. 8, lines 7-27).

28. Regarding claims 21 and 25, **Chow et al.** additionally teaches a database system and article wherein the predetermined type of code corresponding to the first expression includes machine-level code (see disclosure that the procedural and non-procedural parts of the stored procedure are run through compilers, col. 7, line 66 through col. 8, line 6), and wherein the instructions representing the second type expression includes C code (see col. 1, lines 54-62).

29. Regarding claim 22, **Chow et al.** additionally teaches an article wherein the second type of expression comprises a Structured Query Language (SQL) statement (see SQL statements in the stored procedure listed at col. 4, lines 17-32).

30. Regarding claim 24, **Chow et al.** additionally teaches a database system wherein the controller comprises a parsing engine (see disclosure of parser at col. 8, lines 7-27).

31. Regarding claim 26, **Chow et al.** additionally teaches a database system wherein the first type code contains information identifying a type of the expression and a variable and constant used by the expression (see disclosure of variable bind-in, col. 5, line 49 through col. 6, line 36).

32. Regarding claim 28, **Chow et al.** additionally teaches an article wherein the instructions when executed cause the database system to generate the first type code for the expression being one of a conditional expression, assignment expression and dynamic statement, and provide the first type code in the low-level code (see col. 8, lines 7-27).

33. Regarding claim 29, **Chow et al.** additionally teaches an article wherein the instructions when executed cause the database system to generate the second type code which is different from the first type code, and provide the second type code in the low-level code (see disclosure that there are procedural and non-procedural parts of the stored procedure, col. 7, line 66 through col. 8, line 6).

Response to Arguments

34. Applicant's arguments, see amendment, filed 2 September 2004, with respect to the rejection(s)of claim(s) 1-29 under 35 U.S.C. § 102(b) have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made.

Conclusion

35. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Sidik et al. (U.S. Patent 5,675,804) teaches a system for building and linking an interpretive procedure so as to enable a compiled computer program to invoke the procedure.

Wang et al. (U.S. Patent 6,327,629) teaches a universal calling interface for executing a stored procedure in a computer.

PR Newswire ("Oracle Delivers Superior Performance and Scalability with Latest Release of Oracle9i Application Server") is a press release documenting the release of Oracle91 on 5 June 2001.

Naudé et al. ("Oracle9i Release 1 New Features/Upgrade FAQ") teaches that one of the new features of Oracle9i Release 1 is native compilation of PL/SQL programs.

Vijay ("PL/SQL Native Compilation and Performance") teaches the performance gains possible by using native compilation of PL/SQL functions.

Pribyl et al. ("Learning Oracle PL/SQL") teaches the use of native compilation of PL/SQL functions.

Feuerstein et al. ("Oracle PL/SQL Programming") teaches the use of native compilation of PL/SQL functions.

Lamba ("PL/SQL Native Compilation") teaches the use of native compilation of PL/SQL functions.

Schrag ("Native PL/SQL Compilation in Oracle9i") teaches the use of native compilation of PL/SQL functions.

Oracle ("Native Compilation of PL/SQL") teaches the use of native compilation of PL/SQL functions.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Luke S. Wassum whose telephone number is 571-272-4119. The examiner can normally be reached on Monday-Friday 8:30-5:30, alternate Fridays off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John E. Breene can be reached on 571-272-4107. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Luke S. Wassum
Primary Examiner
Art Unit 2167

lsw
30 November 2004